On Resynchronizers for Two-way Transducers

Sougata Bose

LaBRI, Université de Bordeaux

DELTA Paris 2018
Joint work with Anca Muscholl, Gabriele Puppis and Vincent Penelle
The problem

Given two transducers $T_1$ and $T_2$, check if they compute the same relation.
The problem

Given two transducers $T_1$ and $T_2$, check if they compute the same relation.

Classes with known decidable equivalence problem

- Functional 2-way transducers [Culik, Karhumäki, ’87] (PSPACE-Complete)
- 1-way transducers with origin [Filiot et al, ’16] (PSPACE-Complete)
- Streaming string transducers with origin [Bojańczyk et al, ’17]
Equivalence for 2-way with Origin is Decidable
Equivalence for 2-way with Origin is Decidable

Decidability in two steps

- When there are only productive transitions.
- When there are some non-productive transitions.
When there are only productive transitions

Only runs of same **shape** can be origin-equivalent
When there are only productive transitions

- Define a 2NFA $A$ with $Q = Q_1 \times 2^{Q_2}$
When there are only productive transitions

- Define a 2NFA $A$ with $Q = Q_1 \times 2^{Q_2}$
- Track all runs of the same shape with same output for $T_2$
When there are only productive transitions

- Define a 2NFA $A$ with $Q = Q_1 \times 2^{Q_2}$
- Track all runs of the same shape with same output for $T_2$
- $((p, R), a, (p', R'), d) \in \Delta$
- $(p, a, w, p', d)$ in $T_1$
When there are only productive transitions

- Define a 2NFA $A$ with $Q = Q_1 \times 2^{Q_2}$
- Track all runs of the same shape with same output for $T_2$
- $((p, R), a, (p', R'), d) \in \Delta$
- $(p, a, w, p', d)$ in $T_1$
- $r \in R$ with $(r, a, w, r', d)$ in $T_2$, implies $r' \in R'$. 

Theorem \[ L(A) = \emptyset \text{ iff } T_1 \subseteq o T_2 \]
When there are only productive transitions

- Define a 2NFA $A$ with $Q = Q_1 \times 2^{Q_2}$
- Track all runs of the same shape with same output for $T_2$
- $((p, R), a, (p', R'), d) \in \Delta$
- $(p, a, w, p', d)$ in $T_1$
- $r \in R$ with $(r, a, w, r', d)$ in $T_2$, implies $r' \in R'$.
- $F = F_1 \times 2^{Q \setminus F_2}$
When there are only productive transitions

- Define a 2NFA $A$ with $Q = Q_1 \times 2^{Q_2}$
- Track all runs of the same shape with same output for $T_2$
- $((p, R), a, (p', R'), d) \in \Delta$
- $(p, a, w, p', d)$ in $T_1$
- $r \in R$ with $(r, a, w, r', d)$ in $T_2$, implies $r' \in R'$.
- $F = F_1 \times 2^{Q\setminus F_2}$

**Theorem**

$L(A) = \emptyset$ iff $T_1 \subseteq o T_2$
With non-productive transitions

- Put a special symbol $ as output when there is a non-productive transition.
- Equal number of $ means same origin.

Need to eliminate **non-productive loops**
Eliminating non-productive loops

- Annotate the word with information about non-productive loops
- Use this information to get canonical runs
- Check if the annotations are correct
Eliminating non-productive loops

- Annotate the word with information about non-productive loops
- Use this information to get canonical runs
- Check if the annotations are correct

Now we use the previous algorithm
One-way Resynchronizers (Filiot et al '16)

- Transform synchronization language.
- Map input-output pair \((u, v)\) to same pair.
- Change the synchronization.

\[
L_1 = ab \quad L_2 = b^*a
\]
One-way Resynchronizers (Filiot et al ’16)

- Transform synchronization language.
- Map input-output pair \((u, v)\) to same pair.
- Change the synchronization.

\[ L_1 = ab^* \]
\[ L_2 = b^*a \]
One-way Resynchronizers (Filiot et al ’16)

- Transform synchronization language.
- Map input-output pair \((u, v)\) to same pair.
- Change the synchronization.

\[ L_1 = ab^* \quad L_2 = b^*a \]
Objective

Want to compare transducers which generate similar origin graphs.
Objective

Want to compare transducers which generate similar origin graphs

- Transform origin graphs of $T_1$
Objective

Want to compare transducers which generate similar origin graphs

- Transform origin graphs of $T_1$
Resynchronizations

Objective

Want to compare transducers which generate similar origin graphs

- Transform origin graphs of $T_1$

$$G_1 \xrightarrow{R} G_2 \subseteq_o \ ? \ R(G_1)$$
Objective

Want to compare transducers which generate similar origin graphs

- Transform origin graphs of $T_1$

$G_1$ \quad R \quad G_2 \subseteq_o ? \quad R(G_1)$
Resynchronizations

Objective

Want to compare transducers which generate similar origin graphs

- Transform origin graphs of $T_1$

$$G_1 \xrightarrow{R} G_2 \subseteq_o R(G_1)$$

Want $R$ such that $R(G_1)$ is generated by a transducer.
Extending to two-way: Logical Resynchronizers

- A logical way to define transformation origin graphs
Extending to two-way: Logical Resynchronizers

- A logical way to define transformation origin graphs
- MSO-transduction on origin graphs is too powerful
Extending to two-way: Logical Resynchronizers

- A logical way to define transformation origin graphs
- MSO-transduction on origin graphs is too powerful

Our idea

Use MSO formula on the input word
MSO Formula restricted to input

- An MSO formula $\phi(x, y)$ over the input word.
- Two free variables $x$ and $y$. 
MSO Formula restricted to input

- An MSO formula $\phi(x, y)$ over the input word.
- Two free variables $x$ and $y$.
- $R_\phi$ changes origin $x$ to origin $y$. 
MSO Formula restricted to input

- An MSO formula $\phi(x, y)$ over the input word.
- Two free variables $x$ and $y$.
- $R_\phi$ changes origin $x$ to origin $y$.
- $y$ need not be a function of $x$
Restriction on the formula

\[ \phi(x, y) = (y = 1) \]
Restriction on the formula

- $\phi(x, y) = (y = 1)$
- Apply $R\phi$ to graphs generated when a transducer copies the input word by copying every letter.
Restriction on the formula

- \( \phi(x, y) = (y = 1) \)
- Apply \( R_\phi \) to graphs generated when a transducer copies the input word by copying every letter.
- The set of transformed graphs are not generated by any transducer.
Restriction on the formula

- $\phi(x, y) = (y = 1)$
- Apply $R_\phi$ to graphs generated when a transducer copies the input word by copying every letter.
- The set of transformed graphs are not generated by any transducer.

The Restriction

- For a given $y$, the size of the set $\{x \mid \phi(x, y) \text{ is true}\}$ should be bounded.
Restriction on the formula

- $\phi(x, y) = (y = 1)$
- Apply $R_\phi$ to graphs generated when a transducer copies the input word by copying every letter.
- The set of transformed graphs are not generated by any transducer.

The Restriction

- For a given $y$, the size of the set $\{x \mid \phi(x, y) \text{ is true}\}$ should be bounded.

This ensures we can build a transducer generating $R_\phi(G_1)$ from $T_1$. 
Construction of the transducer

- Consider the run of $T_1$ which generates the origin graph
Construction of the transducer

- Consider the run of $T_1$ which generates the origin graph
- We modify this run to obtain the transformed graph
- If output is done at position $x$ in $T_1$, we remember the output to be done, go to $y$ such that $\phi(x, y)$ is true and then do the output at $y$. 
Construction of the transducer

- Consider the run of $T_1$ which generates the origin graph.
- We modify this run to obtain the transformed graph.
- If output is done at position $x$ in $T_1$, we remember the output to be done, go to $y$ such that $\phi(x, y)$ is true and then do the output at $y$.
- Now, we return to the $x$ we came from. This is possible remembering how many $x$’s satisfying $\phi(x, y)$ are situated to the left of $y$. 
Thank You!
Questions?